UE101: Programming & Algorithms

Lecture 1

Code: 1_1.c to 1_5.c, ue101.h

Viraj Kumar

Divecha Centre for Climate Change

Who we are

- Instructors:
  - Prof. Viraj Kumar, DCCC (Programming)
  - Dr. Anand Mishra, CDS (Postdoc.) (Intro. to Algorithms & DS)
  - Prof. Sathish Govindarajan, CSA (Algorithms & Data Structures)
- Teaching Assistants:

Ajinkya Rajput	Ajith S
Anupam Sanghi	Habeeb P
Koti Nishat Saipanmuluk	Mohit Sharma
Shikhar Vashishth	Shivani T
Ullas Aparanji	Vivek V P

Course resources & policies

- Website: http://drona.csa.iisc.ernet.in/~gsat/Course/algoandprog/
- Also: https://canvas.instructure.com/courses/1376618

- Feedback Form (fill by 11:59 PM): https://goo.gl/forms/qWR31aiORJ2Z8wvm2

- 5 Assignments (with vivas) + Labs + Quizzes (~1 per month): 40%
  - **Advice**: Use some of your lab-time to test your assignments on lab machines
- Midterm: 20%
- Final: 40%
- **NOTE**: Plagiarism on assignments

Course resources & p

- Website: http://drona.csa.iisc.ernet.in/~gsat/
- Also: https://canvas.instructure.com/courses

- Feedback Form (fill by 11:59 PM): https://go

- 5 Assignments (with vivas) + Labs + Quizzes
  - **Advice**: Use some of your lab-time to test you
- Midterm:
- Final:
- **NOTE**: Plagiarism on assignments

Note: We have tools to examine all $\binom{120}{2}$ pairs.

Definition of Plagiarism: *Any* attempt to pass off someone else's work as your own, or to **actively** or **passively** assist someone else to do so.

Consequences: *All* students involved will get a **ZERO**. If repeated, we will escalate to the Dean's Office.

Course Objectives

- **Think algorithmically**
  - *Example*: How do you multiply $a_1a_2\ldots a_n$ by $b_1b_2\ldots b_m$ (n- and m-digit numbers)?
  - "…multiply a_i with b_j…" (nested loops)
  - "…add the carry (if it exists)…" (conditionals)
  - This takes mn multiplications. Can we do better? (divide-and-conquer)
- **Leverage data structures**
  - Make algorithms faster by cleverly storing and retrieving data
  - *Example*: Repeatedly insert a new number into a bag or remove the smallest number inserted so far (priority queue)
- **Program**

Key topics and Textbooks

- **C Programming**: Data Types, Functions, Arrays, Pointers, Structures, Recursion
  - Brian W. Kerninghan and Dennis M. Ritchie, The C Programming Language. Prentice Hall of India, 2009
  - Abhiram Ranade, An Introduction to Programming through C++. McGraw Hill Education, 2017
- **Data Structures**: Binary Search Trees, Heaps
  - Robert L. Kruse, Data Structures and Program Design in C. Prentice Hall of India, 2006
  - Mark A. Weiss, Data Structures and Algorithm Analysis in C. Pearson Education India, 2002
- **Algorithms**: Sorting, Graph Traversals, Divide and Conquer, Greedy, Dynamic Programming
  - Steven S. Skiena, The Algorithm Design Manual. Springer, Second Edition, 2008
  - Sanjay Dasgupta, Christos Papadimitriou, and Umesh Vazirani, Algorithms. McGraw Hill, 2017

Levels of programming

- Deep down, the CPU (Central Processing Unit) only understands instructions in binary

```
0011 1001 1101 0001
```

Binary representation of
`cmp ecx, edx`
in x64 (64-bit x86 ISA)

- One level up: Assembly

```
    cmp ecx, edx
    je  same
    ...
same:
    ...
```

- One level up: C, Python, …

```
for (int i = 0, done = 0; !done; ++i) {
    ...
}
```

- There are higher levels (often specialized)
  - *Example*: Programming by example

This course: Programming in C

- Linux (e.g., lab machines): **`gcc`** is probably already installed
- Windows: Download MinGW or MinGW-w64
- Online:
  - Prutor (also for labs)
  - Online Python Tutor (yes, despite the name)

Note: Add security exception

1_1.c

```
int main(){return 0;}
```

Whitespace ignored

```
int main() {
  return 0;
}
```

- Compile: **`gcc 1_1.c`**
  - Execute: **`a.out`** (Linux; sometimes **`./a.out`**) or **`a.exe`** (Windows)
  - Operating System (OS) treats an executable as a **`main`** function that returns an integer (**`int`**)
    - By convention, 0 = "all OK" and non-zero = "error"
    - Linux: **`echo $?`**; Windows: **`echo %ERRORLEVEL%`**
  - The **`main`** function can take command line arguments
- *Note*: Try compiling with **`gcc -S 1_1.c`** and **`gcc -S -O3 1_1.c`**

1_2.c

- "Training wheels" header file:
  - Must be in current folder

```
#include "ue101.h"

int main() {
   int n;
   print_int(n);   // What will
                   // be printed?
   return 0;
}
```

- A variable of type **`int`** named **`n`**
- Comments: **`/* multi-line */`** vs. **`// until end of this line`**
- *Note*: Compile with **`-Wall`**

1_2.c

- "Training wheels" header file:
  - Must be in current folder

```
#include "ue101.h"

int main() {
   int n;
   print_int(n);   /* What will
                      be printed? */
   return 0;
}
```

- A variable of type **`int`** named **`n`**
- Comments: **`/* multi-line */`** vs. **`// until end of this line`**
- *Note*: Compile with **`-Wall`**

1_3.c

- What does this code do?

- Really? *Always*?

```
#include "ue101.h"

int main() {
  int n = read_int();
  print_int(n+1);
  return 0;
}
```

1_4.c

- What does this code do?
- What if **n** = **11**?
- What if **n** = **-7**?

```
#include "ue101.h"

int main() {
  int n;
  repeat(5) {
    n = read_int();
    print_int(n/2);
  }
  return 0;
}
```

1_5.c

- What does this code do?
- What if **`n = 0`**?
- What if **`n = -1`**?

```
#include "ue101.h"

int main() {
  int n = read_int();
  repeat(n) {
    print_int(n);
  }
  return 0;
}
```

Key things we've learned

- Good code-writing style: indentation and comments
- In C, variables are uninitialized (switch on compiler warnings)
- The size of **`int`** (and **`double`**) is limited, hence the range is limited
  - Mathematically incorrect results are possible!
- Division for **`int`** truncates (closer to 0), so **`199/100`** is **`1`**
- Training wheels: **`read_int`**, **`repeat`**, etc.
- Repeating a block 0 or negative times means skipping the block